REMARKS

Applicant thanks Examiner for the detailed review of the application.

Claim Status

Claim 11 has been amended.

Claims 5, 8, and 32 remain cancelled.

Claim Rejections -35 USC § 103(a)

The Office Action states:

> 5. Claims 1-4, 6-7, 9-31 and 33-41 are rejected under 35 U.S.C. 103(a) as being unpatentable over Chamdani et al. (US 2004/0073906 A1) in view of Hoogerbrugge et al. (US 6,615,333 B1).

"The examiner bears the initial burden of factually supporting any prima facie conclusion of obviousness." MPEP § 2142. It is well established that *prima facie* obviousness is only established when three basic criteria are met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *In re Vaeck*, 947 F.2d 488 (Fed. Cir. 1991) (MPEP 2144).

First, applicant respectfully submits that there is no suggestion or motivation to combine Chamdani and Hoogerbrugge. Chamdani only discloses detecting a store from a non-speculative thread that invalidates a speculative thread, i.e. a non-speculative store invalidating a prior speculative load (paragraph 0049).

As an example, assume a prior speculative load reads a current value of 2 from an address A. Yet, a non-speculative store later writes a value of 3 to address A. The previous speculative

load of 2 from address A is incorrect, as according to program order the speculative load should have loaded from address A after the non-speculative write of 3 to address A; instead the speculative load read a 2. However, note the difference between checking if a speculative thread is invalidated due to a non-speculative operation and applicant's claims, which deal with preventing data from a "speculative store," i.e. a store instruction of a speculative thread, from being forwarded to an instruction of a non-speculative thread. In other words, Chamdani discusses the detection of when a speculative thread is invalidated or has loaded a potentially incorrect value, not preventing a speculative thread from potentially providing incorrect data to a currently executing non-speculative threads as in applicant's claims.

Furthermore, Hoogerrugge only contemplates forwarding a load past a store, such that the "prefetched data in the relevant register is replaced by the store," (col 1 lines 48-50). Therefore, Hoggerrugge only focuses on maintaining data validity in an out-of-order machine that executes a forwarding load out-of-order in regards to a store operation, and Hoggerrugge does not disclose or suggest usage of replacing forward loaded data with subsequent store data in the detection of a speculative thread being invalidated by a non-speculative operation, as in Chamdani.

Second, Applicant also respectfully submits that the combination of Chamdani and Hoogerbrugge do not disclose all the elements of applicant's independent claims 1, 11, 21, 26, and 34.

Applicant's claim 1 includes, "**blocker logic** to prevent data associated **with a store instruction of the speculative thread from being forwarded to an instruction of a non-speculative thread**, the blocker logic further to prevent the data from being stored in a memory system," (Claim 1 emphasis added).

In fact, Chamdani's only discloses speculative stores should not irreversibly modify processor state in order to be able to re-execute (paragraph 0031) and a description of not

irreversibly modify processor state through holding such stores in a store buffer (See paragraph 0049). In fact, this disclosure is similar to applicant's description of "category 1 techniques" in paragraph 0025, where complex hardware mechanisms are usually employed to buffer different version of speculatively written memory states…checker logic is employed to determine success of the speculation, and store data is reused if speculation is successful or rolled back in not successful. Note that this described implementation potentially results in "a relatively high cost in terms of hardware complexity," as disclosed in applicant's paragraph 0025. In contrast, applicant's claim 1 prevents speculative store data from being forwarded to non-speculative loads, which potentially results in the elimination of the complex checker logic required by Chamdani, as the possibility of speculative data being forwarded to a non-speculative thread is eliminated.

In addition, The Office Action states Chamdani does not disclose blocker logic to prevent forwarding of data associated with a store instruction to a non-speculative thread. However, applicant respectfully submits that Hoogerrugge also does not disclose this element. Hoogerrugge discusses a general problem for out-of-order processors, where a store instruction precedes a load instruction in program order. However, when dispatched in the processor, the load is forwarded, i.e. is essentially performed out of program order before the store instruction.

For example, assume address A initially holds a previous value of 2, a first store is to write a 3 to address A, and a second subsequent, in program order, load instruction is to load the 3. When executed out-of-order the "forwarded" load reads the previous value of 2 from address A and puts it in register X. Here, when the store is performed, to maintain program order, the store of 3 to A is placed in register X, such that the result maintains the illusion that a store of 3 was performed and then the load of the 3 was performed.

Yet, note the explicit difference between Hoogerrugge and applicant's claim 1. First, applicant's claim 1, is not replacing data of a forwarded load with data from a store operation as in

Hoogerrugge, but in point of fact, is "preventing" data of a speculative store from being forwarded to or replacing a non-speculative operation. In other words, applicant's claim 1 is explicitly preventing data of a store from replacing data to be loaded, while Hoogerrugge only describes replacing data loaded by a forwarded load with store data.

The Office Action states that Hoogerrugge discloses that at the original location of the load instruction (that was forwarded and replaced by store data) a "commit" instruction is added to prevent store instructions after that location from causing a substation with store data (See col. 1 lines 55-58). However, note three extremely important observations regarding this statement. First, the insertion of **a commit instructio**n is used to prevent store data. In other words, a compiler inserted instruction is utilized, i.e. the commit instruction, not **blocker logic** as described in applicant's claim 1. Second, use of preventing subsequent store instructions simply eludes to other store instructions that are later in program order from the original load location and there is no discussion of preventing speculative store data regardless of the program order from being forwarded to a non-speculative operation. Finally, Hoogerrugge's utilization of the commit instruction in the background section is to point out the problem with Chen's implementation, in that, Chen utilizes an additional commit instruction for each forwarded load (See col. 1 lines 62-65), and therefore, explicitly denounces any usage of a commit instruction in this manner. Therefore, nowhere in Hoogerrugge is blocker logic to prevent speculative store data from being forwarded to non-speculative operations, as disclosed in applicant's claim 1.

Similarly, applicant's claim 11 includes, "blocker logic to prevent data associated with a store instruction of a speculative thread from being forwarded to an instruction of a non-speculative thread." As stated above, the combination of Chamdani and Hoogerrugge do not disclose these element of blocker logic to prevent data associated with a store instruction of a speculative thread from being forwarded to an instruction of a non-speculative thread.

Applicant's claim 21 includes, "determining if the load instruction and the in-flight store instruction each originate with a speculative thread… forwarding, if the dependence check is successful, store data associated with the in-flight store instruction to the load instruction; and declining to forward, if the dependence check is not successful, the store data to the load." As stated above, Chamdani only discloses a buffer to buffer speculative stores and a method for detecting invalidation of a speculative thread. However, Chamdani does not disclose the ability to forward speculative data or declining to forward speculative data based on whether a dependence check is successful, i.e. addresses match and each originate from a speculative thread. In addition, Hoogerrugge only focuses on replacing loaded data that was loaded by a forwarded load with in-flight store data. Here note that: (1) Hoogerrugge does not decline to forward store data; and (2) the replacement of loaded data with store data is only conditioned on an address match, not whether the load and store both originate from a speculative thread.

Applicant's clam 26 includes, "marking the cache line as speculative." First, note that Chamdani does not disclose marking a cache line as speculative, but rather only discloses "logic to stall a read from global register **if the thread reading the global register is a speculative thread**," (paragraph 0005). As can be seen, Chamdani only discloses performing a stall if a speculative thread is performing a read, but nowhere in Chamdani is marking of a specific operation or cache line as speculative disclosed. The only inference that may be made from Chamdani's statement is that the system is aware a speculative thread is reading the global register, i.e. then a stall of the read is performed. However, Chamdani makes no disclosure of how the system is aware of the threads speculative nature. In claim 26, applicant provides a specific embodiment, where the speculative nature of data in a cache line is marked as such. The Office Action cites paragraph 0005 in reference to claim 1, which does not disclose any marking logic or method of marking an instruction or cache line, but rather only knowledge of a read operations association with a

speculative thread. Furthermore, in regards to claim 26, only general Figures and related text are cited. Therefore, applicant respectfully requests that the examiner particularly identify where marking of a cache line as speculative is disclosed in Chamdani, as cited in the Office Action. In addition, Hoogerrugge's disclosure is associated with out-of-order execution and makes no mention of speculative or non-speculative nature of instructions or cache lines. Consequently, applicant respectfully submits that the combination does not disclose all of applicant's claim 26 elements.

Similarly, applicant's claim 34 includes, "a storage area to include a speculation identifier (ID) field, the speculation ID field to hold a first value to indicate an associated store instruction is associated with the speculative thread." As discussed above, Chamdani does not disclose any field in any storage area, whether it be blocker logic, a cache line, or a load/store buffer, that identify an associated instruction is speculative. Furthermore, neither Chamdani or Hoogerrugge disclose control logic to prevent data associated with a store instruction from being consumed by a non-speculative thread based on a field that indicates the store instruction is speculative. Note that Hoogerrugge's replacement of loaded data with store data is only conditioned upon and address match, not a value indicating whether an operation/instruction is speculative, while Chamdani does not disclose control logic to prevent data, but rather only detection of speculative thread invalidation.

As a result, applicant respectfully submits that independent claims 1, 11, 21, 26, and 34, as well as their dependent claims, are now in condition for allowance for at least the reasons stated above.

If there are any additional charges, please charge Deposit Account No. 50-0221. The Examiner is invited to contact David P. McAbee at (503) 712-4988 for an interview in light of any further rejection based on Chamdani and/or Hoogerrugge to further discuss the differences between the references and applicant's claims to expedite prosecution.

Respectfully submitted,
Intel Corporation

Dated: October 15, 2008

/David P. McAbee/Reg. No. 58,104
David P. McAbee
Reg. No. 58,104

Intel Corporation
M/S JF3-147
2111 NE 25th Avenue
Hillsboro, OR 97124
Tele – 503-712-4988
Fax – 503-264-1729